ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Nauki Techniczne Nr 143 Informatyka Z. 1

Marek Krętowski[1], Leon Bobrowski[1,2]

GENEROWANIE WIELOWYMIAROWYCH DRZEW DECYZYJNYCH NA PODSTAWIE ZBIORÓW DANYCH

Streszczenie: W pracy przedstawiono nowe metody konstruowania klasyfikatorów o postaci wielowymiarowych (skośnych) drzew decyzyjnych na podstawie zbiorów uczących. Generowane drzewa zawierają w każdym węźle liniową regułę decyzyjną (hiperpłaszczyznę), która dzieli wektory cech docierające do tego węzła na dwa podzbiory. Strategie poszukiwania hiperpłaszczyzny decyzyjnej opierają się na minimalizacji dipolowych funkcji kryterialnych, a jako procedury optymalizujące wykorzystywane są algorytmy wymiany rozwiązań bazowych lub algorytmy ewolucyjne. Aby wyeliminować nadmiarowe cechy, w proces poszukiwania optymalnej kombinacji liniowej w węźle wbudowana została selekcja cech. Dla uniknięcia zbytniego dopasowania do danych treningowych i zwiększenia zdolności do generalizacji, uzyskane drzewo jest przycinane. Przedstawione metody zostały zweryfikowane eksperymentalnie na publicznie dostępnych zbiorach danych i porównane z innymi algorytmami indukującymi drzewa decyzyjne.

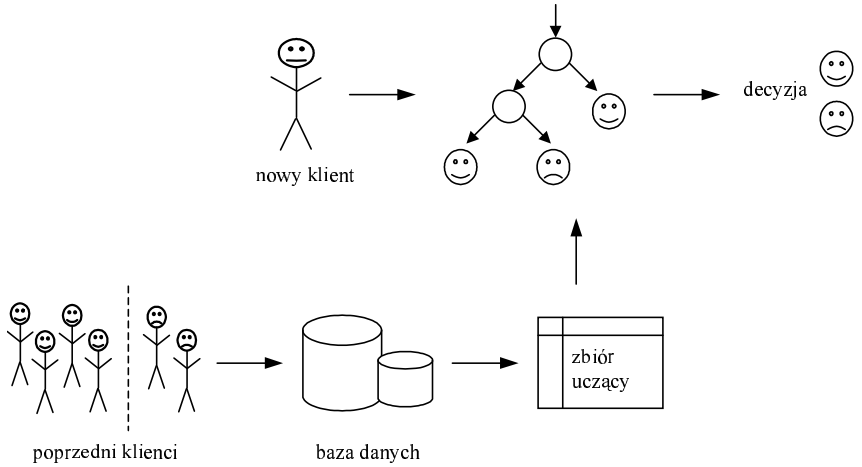
Słowa kluczowe: wielowymiarowe drzewa decyzyjne, liniowa reguła decyzyjna, kryteria dipolowe

1. Wstęp

Z problemami rozpoznawania i klasyfikacji człowiek ma do czynienia w wielu dziedzinach życia, chociaż rzadko je bezpośrednio dostrzega i nazywa. Klasyfikację możemy nieformalnie określić jako czynność polegającą na przypisaniu zjawiskom bądź rzeczom wcześniej sprecyzowanych etykiet (klas, kategorii). W wyniku klasyfikacji dany zbiór obiektów zostaje podzielony na rozłączne grupy. Niekiedy nie zdajemy sobie z tego sprawy, że aby poprawnie klasyfikować obiekty najpierw musimy się nauczyć je rozpoznawać. Czasami dysponujemy wiedzą zdobytą i przekazaną nam przez poprzedników. W takich wypadkach, wystarczy stosować ją w praktyce. Gdy tej wiedzy nie mamy, musimy sami stworzyć reguły

[1] Wydział Informatyki, Politechnika Białostocka, Białystok
[2] Instytut Biocybernetyki i Inżynierii Biomedycznej, PAN, Warszawa

postępowania, opierając się na zdobywanym doświadczeniu. Można przyjąć, że uczenie się na podstawie uprzednio zgromadzonej informacji jest niejako częścią naszego sposobu podejmowania decyzji.



Rys. 1. Schemat wykorzystania metod klasyfikacji (drzew decyzyjnych) w procesie podejmowania decyzji w przykładowej instytucji

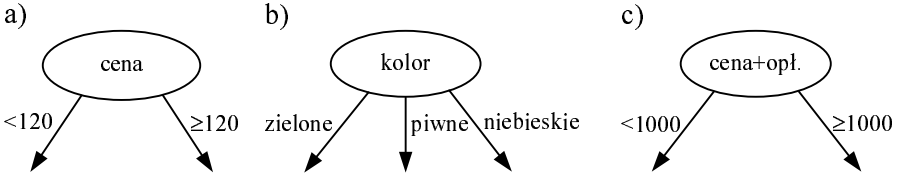
Wraz z nadejściem ery komputerów i ich ciągłym doskonaleniem pojawiła się szansa wykorzystania coraz większej mocy obliczeniowej maszyn we wspomaganiu procesu podejmowania decyzji przez człowieka (rys. 1). Dzięki systemom baz danych mamy możliwość gromadzenia i bezpośredniego dostępu do coraz większej liczby informacji. Powstaje specjalistyczne oprogramowanie, które pozwala na prawie automatyczne przetwarzanie i analizę nawet bardzo dużych zbiorów danych. Uzyskiwane w taki sposób klasyfikatory charakteryzują się wysoką jakością, i same mogą stać się źródłem pożytecznej wiedzy na temat rozpatrywanego zjawiska. Zastosowań jest bardzo wiele i są one niezwykle zróżnicowane: począwszy od ustalenia prawdopodobieństwa zwrócenia pożyczki przez potencjalnego klienta banku, na podstawie wcześniej zrealizowanych pożyczek, po rozpoznawanie komórek nowotworowych na obrazach biomedycznych, czy też wykrywanie nieuprawnionych rozmów w sieciach telefonii komórkowej. Z punktu widzenia metod tworzenia klasyfikatorów wszystkie te problemy są bardzo zbliżone. Zagadnienia tego typu, ze względu na ich bardzo duże znaczenie praktyczne, są przedmiotem intensywnych badań w takich dziedzinach jak statystyka, rozpoznawanie wzorców (ang. *pattern recognition*) czy uczenie maszyn (ang. *machine learning*). Badania zaowocowały m. in. opracowaniem różnego typu klasyfikatorów (np.: w postaci reguł i drzew decyzyjnych czy sieci neuropodobnych) oraz znacznej liczby algo-

rytmów automatycznego tworzenia takich struktur decyzyjnych na podstawie zbiorów danych. Chociaż każda z tych metod ma swoje zalety, wiemy, że nie można zbudować klasyfikatora, optymalnego dla wszystkich sytuacji („no free lunch theorem" [12]). W niniejszej pracy koncentrujemy się na jednym typie klasyfikatora, a mianowicie na drzewach decyzyjnych. Do głównych zalet drzew decyzyjnych możemy zaliczyć przede wszystkim szybkość ich tworzenia i prostotę korzystania z nich oraz łatwość uzasadnienia proponowanych decyzji. W pracy pokazujemy jak wykorzystując metody dipolowe można efektywnie generować wielowymiarowe drzewa decyzyjne na podstawie zbiorów uczących.

Dalsza część pracy jest zorganizowana w następujący sposób. W kolejnym punkcie przedstawione zostanie krótkie wprowadzenie do metody drzew decyzyjnych. Omówiony zostanie podstawowy algorytm generowania drzewa, a także scharakteryzowane różne typy drzew decyzyjnych. Punkt 3 zawiera przegląd istniejących systemów służących do indukcji (głównie wielowymiarowych) drzew decyzyjnych. W punkcie 4 przedstawiono pojęcie dipola i sformułowano postulaty projektowe. Na tej podstawie zostaną zaproponowane metody poszukiwania optymalnych kombinacji liniowych cech w węzłach drzewa decyzyjnego. Metody te obejmują zarówno zdefiniowanie funkcji kryterialnych (służących do oceny jakości proponowanych podziałów) jak i algorytmy optymalizacji tych funkcji. Inne zagadnienia związane z tworzeniem wielowymiarowych drzew decyzyjnych, takie jak np. selekcja cech czy też problem unikania zbytniego dopasowania do danych uczących, zostaną omówione w punkcie 5. Eksperymentalna weryfikacja zaproponowanych algorytmów indukcji drzew znajduje się w punkcie 6. W ostatnim punkcie podsumowujemy oraz przedstawiamy możliwe kierunki prac badawczych.

2. Wprowadzenie do drzew decyzyjnych

Drzewo decyzyjne jest strukturą hierarchiczną, zbudowaną z węzłów i łączących je krawędzi (gałęzi). W strukturze tej jeden węzeł jest wyróżniony i nazywany *korzeniem* (ang. *root node*), a każdy z pozostałych węzłów ma swojego poprzednika w drzewie. Węzły terminalne, czyli takie, które nie posiadają potomków, nazywane są liśćmi (ang. *leaves*). Każdemu liściowi przypisana jest decyzja klasyfikacyjna. W węzłach niebędących liśćmi, zapisane są testy (zwane też regułami podziału). Z każdym możliwym wynikiem testu związana jest gałąź prowadząca do węzła potomnego. W najprostszej sytuacji, teście opartym na pojedynczym atrybucie numerycznym, mamy dwa możliwe wyniki – dwie gałęzie. W przypadku testów opartych na atrybutach nominalnych liczba poddrzew może być większa. Na rysunku 2 przedstawiono przykłady różnych testów w węzłach drzewa.



Rys. 2. Przykładowe testy w węźle drzewa decyzyjnego: (a) jednowymiarowy, binarny, na wartościach rzeczywistych (b) jednowymiarowy, na wartościach nominalnych (c) binarny, wielowymiarowy

Zakładając, że znamy wartości atrybutów nowego obiektu, a nie wiemy, do której klasy należy, możemy przy użyciu drzewa decyzyjnego spróbować przewidzieć jego przynależność do jednej z klas. W tym celu należy wykonać serię testów w kolejnych węzłach drzewa, rozpoczynając od testu zapisanego w korzeniu drzewa, a kończąc na osiągnięciu liścia. Wybór potomka rozpatrywanego węzła drzewa, do którego następuje przejście, dokonywany jest na podstawie wyniku testu zastosowanego do klasyfikowanego obiektu. W momencie, gdy obiekt ten dochodzi do liścia, przypisujemy mu decyzję związaną z tym liściem. Warto zauważyć, że ścieżka od korzenia do liścia odpowiada pojedynczej regule decyzyjnej, z testami połączonymi koniunkcją po jej lewej stronie i decyzją związaną z liściem po stronie prawej.

W zasadzie wszystkie algorytmy generowania drzew decyzyjnych na podstawie zbioru danych opierają się na powszechnie stosowanej w informatyce metodzie znajdującej rozwiązanie problemu poprzez podział go na podproblemy (ang. *divide and conquer*). W odniesieniu do drzew, rozpoczynając od korzenia, znajdowany jest test i na podstawie jego wyników tworzone są węzły potomne oraz rozdzielany do nich jest zbiór uczący. Następnie ta sama procedura podziału stosowana jest rekurencyjnie w węzłach potomnych, aż do momentu, kiedy osiągnięty zostaje warunek stopu i tworzony jest liść. Poniżej przedstawiono podstawowy algorytm indukcji drzewa w postaci funkcji w pseudokodzie:

```
void InduceDecTree(DTNode *pN)
{
    ...                 // sprawdzany jest warunek stopu
    if (!Stop(pN))      // np. czy w ogóle warto dzielić

    if (FindTest(pN))   // poszukiwanie testu i o ile zakończone
                        // pomyślnie zwracane true
        {
        nOutcome=SplitNode(pN);    //liczba możliwych wyników testu
                                   //a tym samym gałęzi (podrzew)
            for (int i=0; i<nOutcome; i++)
```

```
            InduceDecTree(pN->GetOutcome(i));
        }
        ...
        // o ile nie znaleziono testu w pN oznaczamy jako liść
        // w przeciwnym przypadku jako węzeł
    }
```

Należy zwrócić uwagę, że opisana powyżej zachłanna (ang. *greedy*) strategia indukcji drzewa od korzenia do liści (ang. *top-down*), jak każda metoda heurystyczna, nie gwarantuje otrzymania optymalnego drzewa. Tym niemniej jest prosta i szybka, a co najważniejsze, uzyskiwane dzięki niej klasyfikatory charakteryzują się dobrą jakością klasyfikacji i małymi rozmiarami.

Podstawowe problemy, które musimy rozwiązać proponując metodę generowania drzewa decyzyjnego [9], to przede wszystkim:

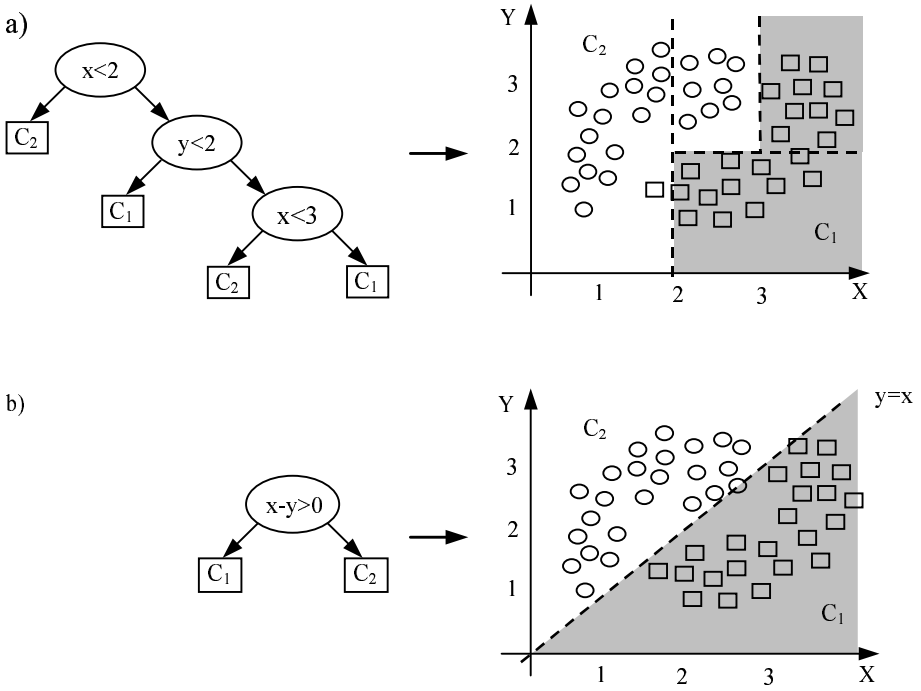
a) sposób wyboru testu w węźle drzewa – `FindTest(pN)`,
b) określenie kiedy przerwać rekurencyjny podział – `Stop(pN)`,
c) przypisanie decyzji liściowi (zwykle stosowana jest reguła większościowa).

Ad a) Najczęściej stosowana jest metoda wyboru optymalnego testu, polegająca na maksymalnej redukcji zanieczyszczenia (ang. *impurity*) węzła w jego potencjalnych potomkach. W przypadku najprostszych postaci testów możliwe jest nawet sprawdzanie wszystkich znaczących podziałów. Jako miary zanieczyszczenia wykorzystywane są najczęściej: *indeks Gini* [9] lub *entropia* [21].
Ad b) Najbardziej naturalną przesłanką do zatrzymania algorytmu jest otrzymanie węzła zawierającego tylko obiekty z jednej klasy. Zastosowanie takiego kryterium pozwala uzyskać, o ile zbiór uczący nie jest sprzeczny, drzewo poprawnie klasyfikujące wszystkie obiekty ze zbioru uczącego. Niestety, powoduje to często nadmierne dopasowanie się do danych treningowych. Do tego zagadnienia powrócimy w dalszej części pracy.

W zależności od liczby cech wykorzystywanych przez regułę decyzyjną w węźle możemy wyróżnić dwie rodziny drzew decyzyjnych: jednowymiarowe lub wielowymiarowe. Drzewa jednowymiarowe wykorzystują w każdym teście tylko pojedynczą cechę, natomiast wielowymiarowe mogą opierać się na większej liczbie zmiennych w poszczególnych węzłach. Szczególnym przypadkiem drzew wielowymiarowych są tak zwane drzewa skośne (ang. *oblique*), w których reguła decyzyjna ma postać liniowej kombinacji cech.

Z geometrycznego punktu widzenia jednowymiarowe testy związane są z hiperpłaszczyznami równoległymi do osi (ang. *axis-parallel*) w przypadku liniowych testów wielowymiarowych, orientacja hiperpłaszczyzn nie jest praktycznie ograniczona. Drzewa jednowymiarowe znalazły wiele praktycznych zastosowań, głównie dzięki dostępności komercyjnych systemów służących do ich generowania oraz

dzięki szybkości tworzenie i łatwości interpretacji uzyskiwanych klasyfikatorów. Drzewa wielowymiarowe oferują znacznie większe możliwości, choć – niestety – ich indukcja może wymagać więcej czasu (algorytmy poszukiwania testów w węzłach są zwykle bardziej skomplikowane) interpretacja uzyskanego drzewa może być nieco trudniejsza. Na rysunku 3 przedstawiona została sytuacja, w której klasyfikator w postaci skośnego drzewa decyzyjnego jest naturalnym i optymalnym rozwiązaniem.



Rys. 3. Przykładowe drzewa wygenerowane na podstawie tego samego dwuklasowego zbioru danych: (a) jednowymiarowe, (b) wielowymiarowe (skośne)

3. Przegląd metod konstrukcji drzew decyzyjnych

Zaproponowano już wiele systemów generujących drzewa decyzyjne na podstawie zbiorów uczących [19]. Początkowo interesowano się jedynie drzewami jednowymiarowymi. Do najbardziej znanych systemów należy z pewnością zaliczyć ID3 [21], zaproponowany przez Quinlana. System ten ewoluował przez lata i jego publicznie dostępna wersja, pod nazwą C4.5 [22], jest powszechnie wykorzystywana we wszelkiego typu porównaniach. Spośród innych algorytmów warto

wymienić CHAID (*Chi-squared Automatic Interaction Detection*) [15], który może być stosowany tylko w przypadku danych nominalnych; do wyboru podziału w węźle wykorzystuje się statystykę chi-kwadrat.

Algorytmy stosowane w przypadku wielowymiarowych drzew decyzyjnych są ze swej natury bardziej złożone niż wykorzystywane w drzewach jednowymiarowych. Dlatego też w początkowym okresie ich rozwój i praktyczne zastosowanie było hamowane przede wszystkim z powodu ograniczonych możliwości komputerów. W ostatnim jednak czasie, dzięki wzrostowi mocy obliczeniowej, drzewa wielowymiarowe zasługują na coraz większą uwagę. W tej chwili mogą być z powodzeniem stosowane nawet w przypadku dużych zbiorów danych (zawierających tysiące obiektów).

Po raz pierwszy możliwość wykorzystania wielowymiarowych testów podczas konstrukcji drzewa została opisana w powszechnie znanej książce Breimana i in. [9]. Autorzy opisują w niej system CART (*Classification and Regression Tree*), który, oprócz standardowych podziałów wykorzystujących pojedyncze zmienne, jest w stanie m. in. szukać testów w postaci liniowej kombinacji cech. Do poszukiwania hiperpłaszczyzny użyte są tylko cechy numeryczne, a wagi są dodatkowo znormalizowane (suma kwadratów wag jest równa 1). Zaproponowany algorytm w kolejnych krokach próbuje modyfikować wagi pojedynczych atrybutów, optymalizując miarę zanieczyszczenia. Następnie w celu uproszczenia tekstu atrybuty są usuwane z kombinacji liniowej, o ile strata przez to spowodowana nie przekracza zadanego procentowego progu. Po znalezieniu optymalnej hiperpłaszczyzny, miara zanieczyszczenia jest porównywana z miarą odpowiadającą najlepszemu podziałowi jednowymiarowemu. Na tej podstawie wybierany jest lepszy test, przy czym preferowane są testy oparte na pojedynczej zmiennej.

Metoda zaproponowana przez Breimana *i in.* jest w pełni deterministyczna i może wpadać w lokalne minima. Murthy *i in.* [18] zaproponowali system OC1 (*Oblique Classifier 1*), rozszerzający metodę poszukiwania liniowych kombinacji systemu CART poprzez wprowadzenie randomizacji. W momencie gdy algorytm zatrzymuje się (być może w lokalnym minimum), system próbuje dodawać losowe wektory do aktualnie optymalnej hiperpłaszczyzny i, jeżeli uzyskana kombinacja liniowa jest lepsza, kontynuuje poszukiwania. Inną możliwością jest kilkukrotny start z losowo wybranych miejsc, różnych od użytego w pierwszej próbie optymalnego testu jednowymiarowego. Wyniki uzyskiwane przy użyciu tak rozszerzonej metody uległy znaczącej poprawie i system został wykorzystany z powodzeniem m. in. w diagnostyce raka piersi.

Chai *et al.* [11] zaproponowali metodę tworzenia liniowo-odcinkowego klasyfikatora w postaci binarnego drzewa (BTGA, *Binary Tree-Genetic Algorithm*) z liniową regułą decyzyjną w każdym węźle. Do poszukiwania optymalnej hiperpłaszczyzny, maksymalnie redukującej zanieczyszczenie, wykorzystany został

standardowy algorytm genetyczny. Zmodyfikowana nieznacznie metoda została wykorzystana z powodzeniem w diagnostyce raka szyjki macicy do klasyfikacji komórek na wymazach cytologicznych.

W pracy [14] Gama i Brazdil przedstawili system *Ltree*, który rozszerza algorytm generowania drzewa decyzyjnego poprzez wykorzystanie konstruktywnej indukcji. W każdym węźle nowe atrybuty tworzone są poprzez rzutowanie wektorów cech na hiperpłaszczyznę wyznaczoną przez liniową regułę dyskryminacyjną. W ten sposób utworzone cechy są następnie propagowane w dół drzewa i mogą być wykorzystane w tworzeniu kolejnych cech.

Większość algorytmów indukcji drzew decyzyjnych wybiera podział w węźle optymalizując, w ten czy inny sposób zdefiniowaną, miarę zanieczyszczenia. Nie jest to jednak jedyna możliwość. Przykładowo w pracy [20] autorzy argumentują, że w przypadku drzew skośnych lepsze rezultaty można osiągnąć stosując do oceny potencjalnych podziałów w węźle miarę bazującą na liniowej separowalności danych. Ponieważ zaproponowana miara wykorzystuje liniową separowalność podwęzłów, może być traktowana jako rozszerzenie tradycyjnej zachłannej strategii konstrukcji drzewa o sprawdzenie „krok do przodu". Podejście to może być zastosowane tylko do problemów dwuklasowych.

Przedstawione powyżej systemy wykorzystują pojedynczą hiperpłaszczyznę w węźle drzewa. Istnieją również drzewa wielowymiarowe oparte na *maszynie liniowej* (ang. *linear machine*) [12], która jest zbiorem działających wspólnie liniowych funkcji dyskryminacyjnych. Najbardziej znanym takim drzewem jest LMDT (*Linear Machine Decision Tree*), zaproponowany przez Utgoff i Brodley [23]. W przeciwieństwie do wcześniej omówionych binarnych drzew, w tym przypadku liczba poddrzew zależna jest od liczby klas. Do uczenia maszyny liniowej w każdym węźle stosowany jest algorytm *thermal training* zbliżony do algorytmu symulowanego wyżarzania (ang. *simulated annealing*). LMDT wyposażony jest również w mechanizmy selekcji cech.

4. Poszukiwanie hiperpłaszczyzny w węźle drzewa

W tym paragrafie zostanie przedstawiony sposób poszukiwania liniowej reguły decyzyjnej w pojedynczym węźle drzewa na podstawie danych ze zbioru uczącego, które dotarły do rozpatrywanego węzła. Omówione zostaną miary jakości podziału (funkcje kryterialne) oparte na pojęciu dipoli oraz opisane metody wykorzystywane do ich optymalizacji.

Zbiór uczący C składa się z M obiektów, którym dla porządku przypisujemy numery w zbiorze. Każdy obiekt O^i opisany jest przez N – wymiarowy wektor cech $\mathbf{x}^i = \left[x_1^i,...,x_N^i\right]^T$, ($x_j^i \in R^1$). W przypadku cech nominalnych (takich jak przykładowo *kolor oczu* czy *kraj zamieszkania*) niezbędne jest wcześniejsze zakodowanie tego typu informacji[3]. Dodatkowo każdy obiekt należy do (jednej z K klas) klasy ω_k ($k = 1,..., K$). W podzbiorze uczącym C_k zgrupowane są wszystkie wektory cech $\mathrm{x}^i(k)$, odpowiadające obiektowi O^i, należące do klasy ω_k (C_k zawiera m_k wektorów cech $\mathbf{x}^i(k)$).

Przestrzeń cech może być podzielona na dwie podprzestrzenie przy użyciu hiperpłaszczyzny $H(\mathbf{w},\theta)$ będącej kombinacją liniową cech:

$$H(\mathbf{w},\theta) = \{\mathbf{x}:\langle \mathbf{w},\mathbf{x}\rangle = w_1x_1 + w_2x_2 + ... + w_Nx_N = \theta\}, \tag{1}$$

gdzie $\mathbf{w}$ jest N-wymiarowym wektorem wag, θ – progiem, a $\langle \mathbf{w},\mathbf{x}\rangle$ jest iloczynem skalarnym. Mówimy, że wektor cech $\mathbf{x}^i$ leży po *dodatniej* (*ujemnej*) stronie hiperpłaszczyzny $H(\mathbf{w},\theta)$ jeżeli:

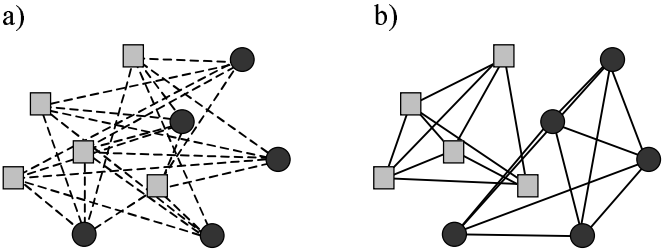
$$\langle \mathbf{w},\mathbf{x}^i\rangle > \theta \qquad (\langle \mathbf{w},\mathbf{x}^i\rangle < \theta). \tag{2}$$

Aby uprościć zapis, wprowadźmy tak zwaną rozszerzoną przestrzeń cech, w której wektor cech ma postać $\mathbf{y} = [1, x_1,..., x_N]^T$, rozszerzony wektor wag przedstawia się następująco $\mathbf{v} = [-\theta, w_1,..., w_N]^T$, natomiast $H(\mathbf{v}) = \{\mathbf{y} : \langle \mathbf{v},\mathbf{y}\rangle = 0\}$ jest hiperpłaszczyzną.

4.1. Ocena jakości hiperpłaszczyzny – funkcja kryterialna

Podstawowym pojęciem, od którego przedstawienia rozpoczniemy, jest *dipol*, czyli para $(\mathbf{y}^i,\mathbf{y}^j)$ różnych wektorów cech [3]. Wyróżniamy dwa typy dipoli: *czyste* i *mieszane*. Mówimy, że dipol jest czysty, jeżeli wektory cech tworzące go należą do tego samego podzbioru uczącego C_k. Z dipolem mieszanym mamy natomiast do czynienia w sytuacji, gdy obiekty tworzące dipol mają przypisane różne decyzje. Na rysunku 4 zilustrowano typy dipoli na przykładzie 10-elementowego zbioru danych, natomiast w tabeli 1 przedstawione zostały liczności poszczególnych typów dipoli.

[3] Istnieje wiele metod kodowania informacji nominalnych (symbolicznych) w postaci numerycznej. Jedną z najprostszych jest stworzenie dodatkowych binarnych atrybutów odpowiadających poszczególnym kategoriom.



Rys. 4. Wszystkie dipole mieszane (a) oraz czyste (b), utworzone z wektorów cech (punktów) należących do przykładowego 10-elementowego zbioru danych

Tabela 1

Liczby dipoli poszczególnych typów

Liczba czystych	$D_c = \frac{1}{2} \cdot \sum_{i=1}^{K} m_i \cdot (m_i - 1)$
Liczba mieszanych	$D_m = \frac{1}{2} \cdot \sum_{i=1}^{K} m_i \cdot (M - m_i)$
Razem (wszystkich) dipoli	$\frac{1}{2} \cdot M \cdot (M - 1)$

Hiperpłaszczyzna $H(\mathbf{v})$ przecina dipol $(\mathbf{y}^i, \mathbf{y}^j)$ jeżeli wektory cech go tworzące znajdują się po jej przeciwnych stronach, czyli:

$$\left\langle \mathbf{v}, \mathbf{y}^i \right\rangle \cdot \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle < 0 \tag{3}$$

Z punktu widzenia wykorzystania liniowej reguły decyzyjnej do podziału wektorów cech w węźle drzewa, hiperpłaszczyzna powinna przecinać możliwie dużo dipoli mieszanych. W idealnym przypadku, jeżeli wszystkie dipole mieszane są przecięte, mamy do czynienia z liniowym odseparowaniem podzbiorów uczących (przypadek 2-klasowy). Z drugiej strony powinniśmy się starać unikać przecinania dipoli czystych. Przecięcie ich oznacza bowiem, że wektory cech z tej samej klasy skierowane zostaną do różnych poddrzew.

Bazując na powyższym rozważaniu można zaproponować [6] prostą miarę jakości podziału (funkcję kryterialną) postaci:

$$\Psi_1(\mathbf{v}) = \alpha_c \cdot \frac{d_c}{D_c} + \alpha_m \cdot \frac{d_m}{D_m} \tag{4}$$

gdzie:

d_c – liczba dipoli czystych, które zostały przecięte przez $H(\mathbf{v})$,

d_m – liczba dipoli mieszanych, które nie zostały przecięte przez $H(\mathbf{v})$,

α_c, α_m – relatywny wpływ (cena) poszczególnych rodzajów dipoli na wartość funkcji.

Należy zwrócić uwagę na fakt, że cena α_m związana z dipolami mieszanymi powinna być znacząco większa niż cena α_c odnosząca się do dipoli czystych. Podstawowym celem jest przecinanie dipoli mieszanych. W drugiej kolejności staramy się unikać przecinania dipoli czystych. W przeciwnym razie możemy doprowadzić do sytuacji, w której żaden dipol nie będzie przecięty, a takie rozwiązanie jest zwykle bezużyteczne. Poszukiwanie optymalnych hiperpłaszczyzn $H(\mathbf{v})$ opiera się na minimalizacji funkcji kryterialnej $\Psi_1(\mathbf{v})$ (4).

Jeżeli bierzemy pod uwagę wszystkie dipole i jeżeli przez $p(k)$ oznaczymy obiekty z klasy ω_k położone po pozytywnej (dodatniej) stronie hiperpłaszczyzny $H(\mathbf{v})$, a przez $n(k)$ odpowiednio po stronie negatywnej, to d_m i d_c możemy obliczyć w następujący sposób:

$$d_m = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \left(p(i) \cdot p(j) + n(i) \cdot n(j) \right), \; d_c = \sum_{i=1}^{K} p(i) \cdot n(i). \tag{5}$$

Ponieważ funkcja Ψ_1 ma wiele minimów lokalnych, jej minimalizacja nie jest prostym zadaniem optymalizacyjnym. W przypadku takich właśnie funkcji dobrze sprawdzają się metody takie jak *algorytmy ewolucyjne* (*genetyczne*) [17]. Niestety, nie należą one do najszybszych metod i, zwłaszcza w przypadku dużych zbiorów danych, tworzenie drzewa decyzyjnego może wymagać sporo czasu.

Alternatywne podejście do poszukiwania hiperpłaszczyzny zainspirowane zostało algorytmami uczenia *Perceptronu* [12], które w przypadku liniowej separowalności zbioru (2 klasy) dają rozwiązania optymalne. W klasycznej funkcji perceptronowej z każdym błędnie sklasyfikowanym przykładem wiążemy karę w postaci iloczynu skalarnego, która może być traktowana jako odległość obiektu od hiperpłaszczyzny (w przypadku unormowania wektora wag). Minimalizowana funkcja kryterialna jest sumą tych kar. Dodatkowo wprowadzenie marginesów (ang. *margin*) pozwala znajdować rozwiązania bardziej „stabilne", a funkcja kryterialna pozostaje nadal wypukła i odcinkowo-liniowa (ang. *piecewise-linear*). Do optymalizacji funkcji tego typu zostały zaproponowane efektywne algorytmy, takie jak algorytmy wymiany rozwiązań bazowych ([2], [8]). Aby móc wykorzystać zalety tych technik, dipolowa funkcja kryterialna również musi mieć analogiczne właściwości.

W naszym przypadku z wektorem cech $\mathbf{y}^j$ możemy związać dwa typy składowych funkcji kary: „pozytywną”

$$\varphi_j^+(\mathbf{v}) = \begin{cases} \delta^j - \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle < \delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \geq \delta^j \end{cases}, \tag{6}$$

lub „negatywną”

$$\varphi_j^-(\mathbf{v}) = \begin{cases} \delta^j + \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle > -\delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \leq -\delta^j \end{cases}, \tag{7}$$

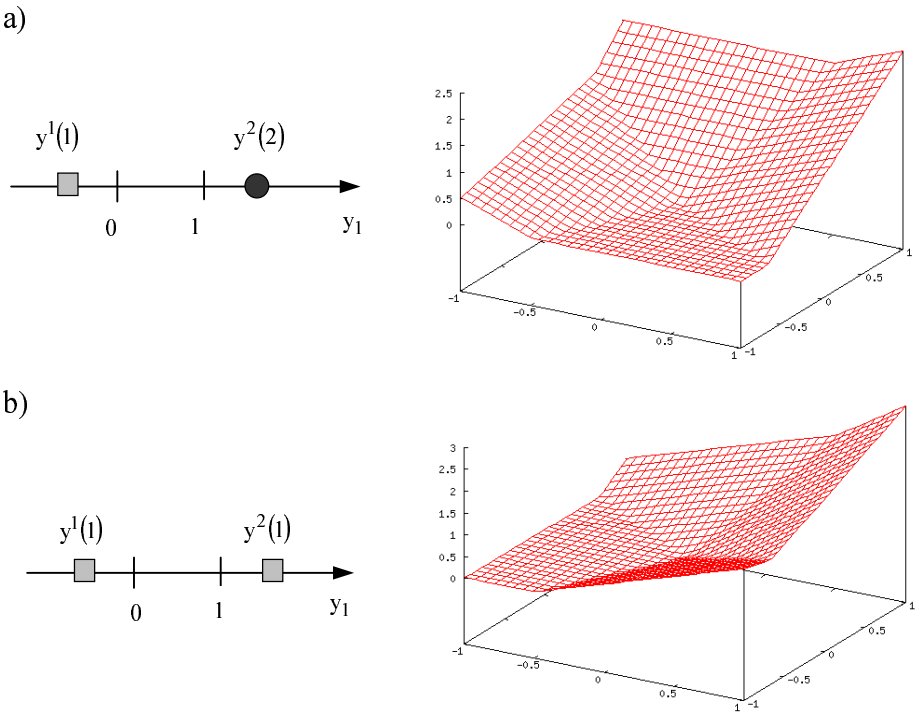
gdzie δ^j ($\delta^j \geq 0$) jest marginesem, którego wartość może być różna dla różnych przypadków (we wszystkich prezentowanych w pracy eksperymentach miała wartość 1.0). Nazwy funkcji kary odpowiadają stronom w stosunku do poszukiwanej hiperpłaszczyzny $H(\mathbf{v})$, po których powinien się znaleźć wektor cech $\mathbf{y}^j$ w wyniku minimalizacji funkcji kary. Inaczej mówiąc, w przestrzeni wag wykorzystanie pozytywnej funkcji kary powoduje spychanie wektora wag na dodatnią stronę hiperpłaszczyzny związanej z wektorem cech. Opierając się na składowych funkcjach kary można łatwo osiągnąć wymagany efekt w odniesieniu do dipoli. I tak, aby wymusić przecinanie dipola mieszanego $\left(\mathbf{y}^i, \mathbf{y}^j\right)$ przez $H(\mathbf{v})$, należy związać z obiektami tworzącymi dipol przeciwne funkcje kary, np.: ujemną $\varphi_j^-(\mathbf{v})$ oraz dodatnią $\varphi_i^+(\mathbf{v})$ (lub odwrotnie). Możemy zdefiniować funkcję kary związaną z dipolem mieszanym jako sumę odpowiednich funkcji:

$$\varphi_{ij}^m(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^-(\mathbf{v}) \qquad (\text{lub } \varphi_{ij}^m(\mathbf{v}) = \varphi_i^-(\mathbf{v}) + \varphi_j^+(\mathbf{v})) \tag{8}$$

W przypadku dipola czystego, aby uniknąć jego przecięcia, wiążemy z obiektami tworzącymi ten dipol funkcje kary tego samego znaku, dzięki czemu obiekty te powinny się znaleźć po tej samej stronie hiperpłaszczyzny. Funkcja kary związana z czystym dipolem przybiera postać następującą:

$$\varphi_{ij}^c(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^+(\mathbf{v}) \qquad (\text{lub } \varphi_{ij}^c(\mathbf{v}) = \varphi_i^-(\mathbf{v}) + \varphi_j^-(\mathbf{v})) \tag{9}$$

Aby móc lepiej wyobrazić wygląd funkcji kary, rozważmy pojedyncze dipole (mieszany i czysty) w jednowymiarowej przestrzeni cech. Na rysunku 5 przedstawione zostały odpowiadające im składowe funkcje kary w przestrzeni wag.



Rysunek 5. Pojedyncze dipole (mieszany – a oraz czysty – b) w 1-wymiarowej przestrzeni cech oraz wykresy składowych funkcji kary $\varphi_{12}^{m}(\mathrm{v})$ oraz $\varphi_{12}^{c}(\mathrm{v})$, związanych z danym typami dipoli, przedstawione w 2-wymiarowej przestrzeni wag

Wybór odpowiednich postaci funkcji kary, związanych z danym dipolem, nazywany jest *orientowaniem dipola* [4]. Aby zbytnio nie komplikować rozważań, można przyjąć, że orientacja dipoli jest dowolna, ale ustalona.

Dipolowa funkcja kryterialna [7] jest ważoną sumą funkcji kar związanych z poszczególnymi dipolami:

$$\Psi_2(\mathbf{v}) = \frac{1}{D_m} \cdot \sum_{\left(y^i, y^j\right) \in I^m} \alpha_{ij} \cdot \varphi_{ij}^{m}(\mathbf{v}) + \frac{1}{D_c} \cdot \sum_{\left(y^i, y^j\right) \in I^c} \alpha_{ij} \cdot \varphi_{ij}^{c}(\mathbf{v}) \qquad (10)$$

gdzie:

α_{ij} – relatywny wpływ (cena) dipola $\left(\mathbf{y}^i, \mathbf{y}^j\right)$ na wartość funkcji kryterialnej,

I^m (I^c) – zbiór dipoli mieszanych (czystych).

Warto zauważyć, że o ile w funkcji $\Psi_1(\mathbf{v})$ wszystkie dipole jednego typu były traktowane w ten sam sposób (miały przypisany ten sam wpływ α_m lub α_c), o tyle w przypadku funkcji $\Psi_2(\mathbf{v})$ każdy dipol może być traktowany niezależnie (cena jest bowiem związana z pojedynczym dipolem). Rodzi to możliwość proponowania różnorodnych strategii wyceniania dipoli i przez to wpływania na kształt funkcji kryterialnej a tym samym na optymalną hiperpłaszczyznę. Przykładowo można różnicować ceny dipoli, wykorzystując ich długości. W przedstawionych w tej pracy eksperymentach zastosowana została najprostsza z możliwych strategii ze stałymi cenami.

Dipolowa funkcja kryterialna (z ustaloną orientacją dipoli) jest, oczywiście, wypukła i odcinkowo-liniowa (jako ważona suma takich właśnie funkcji). Dzięki temu do jej minimalizacji, tak jak wspomniano wcześniej, możemy wykorzystać algorytmy wymiany rozwiązań bazowych. W rzeczywistości wykorzystana metoda jest kombinacją powyższych algorytmów i prostego algorytmu ustalania orientacji dipoli.

W pewnych sytuacjach (zwłaszcza gdy liczba klas jest większa niż 2) bardzo dobre rezultaty mogą być uzyskane poprzez minimalizację *kryterium rangowego* [5], które może być traktowane jako szczególny przypadek kryterium dipolowego. Celem kryterium rangowego jest odseparowanie obiektów z wyróżnionej klasy od wszystkich pozostałych obiektów. Można to osiągnąć poprzez przypisanie obiektom z klasy wyróżnionej składowych funkcji kary pozytywnych, natomiast pozostałym obiektom – negatywnych. W ten sposób dążymy do sytuacji, w której możliwie wiele obiektów z klasy wyróżnionej znajdzie się po dodatniej stronie hiperpłaszczyzny; pozostałe obiekty usytuowane są po stronie ujemnej. Warto zauważyć, że z punktu widzenia zastosowania metody dipolowej kryterium rangowe oznacza wykorzystanie tylko mieszanych dipoli, stworzonych z obiektów klasy wyróżnionej i pozostałych obiektów, przy czym orientacja jest taka sama dla wszystkich dipoli.

4.2. Metody optymalizacji funkcji kryterialnej

W przypadku optymalizacji funkcji, o której wiemy, że jest odcinkami-stała (ang. *piecewise constant*) i zapewne posiada minima lokalne, niezbędne jest zastosowanie metod, które przede wszystkim są w stanie unikać rozwiązań suboptymalnych. Szybkość działania jest w tej sytuacji czynnikiem mniej istotnym. Takimi technikami z pewnością są szeroko rozumiane algorytmy genetyczne. Do minimalizacji funkcji kryterialnej Ψ_1 zaprojektowany został wyspecjalizowany algorytm ewolucyjny, dzięki czemu udało się poprawić efektywność takiego podejścia. W przypadku funkcji kryterialnej Ψ_2 zastosowane zostało odmienne rozwiązanie.

Już sama funkcja zaprojektowana została z myślą o konkretnej technice optymalizacyjnej (algorytmach wymiany rozwiązań bazowych) i w związku z tym posiada określone własności: jest wypukła i odcinkowo-liniowa (ang. *piecewise linear*).

4.2.1. Algorytmy wymiany rozwiązań bazowych

Algorytmy wymiany rozwiązań bazowych ([2], [8]) są efektywnymi iteracyjnymi metodami poszukiwania minimum wypukłych, odcinkowo-liniowych funkcji kryterialnych, utworzonych na podstawie zbioru danych. Są to metody bliskie programowaniu liniowemu, a ich siła bierze się między innymi z wykorzystania specyfiki optymalizowanej funkcji. Podstawowa obserwacja niezbędna do zrozumienia zasady działania algorytmów wskazuje, że hiperpłaszczyzny związane z wektorami cech (przesuniętymi odpowiednio o $+\delta$ lub $-\delta$) rozbijają przestrzeń wag na wypukłe obszary, w których funkcja kryterialna jest liniowa. Obszary te nazywane są obszarami *korekcyjnymi*, a ich wierzchołki wyznaczone są przez punkty przecięcia granicznych hiperpłaszczyzn (N+1 hiperpłaszczyzn w wierzchołku niezdegenerowanym). Z podstawowych twierdzeń programowania liniowego wiemy, że optimum funkcji rozpatrywanego typu zostanie osiągnięte w jednym z wierzchołków obszaru korekcyjnego. Jeśli oprzemy się na powyższym, nasuwa się idea algorytmu polegająca na przemieszczaniu się wzdłuż krawędzi (wyznaczonych przez hiperpłaszczyzny) od jednego wierzchołka do drugiego. Kolejne odwiedzane wierzchołki powinny mieć nie większą wartość funkcji kryterialnej i, o ile nie będziemy odwiedzać ponownie tych samych wierzchołków, mamy zagwarantowane odnalezienie minimum. Poszczególne algorytmy różnią się między sobą sposobem wyboru drogi, po której podążamy, aby osiągnąć optimum. W dalszym opisie skoncentrujemy się na strategii *krawędzi największego spadku*.

Współrzędne każdego wierzchołka $\mathbf{v}(n)$ obszaru korekcyjnego w n-tym kroku algorytmu można wyznaczyć rozwiązując układ N+1 równań liniowych, który w postaci macierzowej wygląda następująco:

$$\mathbf{B}(n)\cdot\mathbf{v}(n)=\boldsymbol{\delta}(n) \tag{11}$$

gdzie $\mathbf{B}(n)$ jest macierzą, zwaną n-tą *bazą*, której wierszami jest N+1 niezależnych wektorów cech $\mathbf{y}^j$, związanych z hiperpłaszczyznami przecinającymi się w wierzchołku $\mathbf{v}(n)$. Składowe wektora marginesów $\boldsymbol{\delta}(n)$ mogą przyjmować odpowiednio wartości $+\delta^j$ lub $-\delta^j$ zgodnie z położeniem hiperpłaszczyzny związanej z danym wektorem cech $\mathbf{y}^j$ w bazie $\mathbf{B}(n)$. Tak więc baza $\mathbf{B}(n)$ oraz wektor $\boldsymbol{\delta}(n)$ jednoznacznie wyznaczają wierzchołek $\mathbf{v}(n)$:

$$\mathbf{v}(n)=\mathbf{B}^{-1}(n)\cdot\boldsymbol{\delta}(n) \tag{12}$$

W każdym kroku algorytmu następuje przejście z $\mathbf{v}(n)$ do następnego wierzchołka $\mathbf{v}(n+1)$, co jest równoznaczne ze zmianą bazy $\mathbf{B}(n)$ na $\mathbf{B}(n+1)$ oraz $\boldsymbol{\delta}(n)$ na $\boldsymbol{\delta}(n+1)$. Jeśli dwie kolejne bazy powstają przez wymianę pojedynczego wektora cech, to do wyliczenia $\mathbf{B}^{-1}(n+1)$ na podstawie $\mathbf{B}^{-1}(n)$ wystarczy zastosować *transformację Gaussa-Jordana*.

Aby wybrać kierunek przechodzenia pomiędzy wierzchołkami obszarów korekcyjnych, wykorzystujemy średni wektor korekcji $\mathbf{k}(\mathbf{v}) = -\nabla\Psi_2(\mathbf{v})$. Gradient funkcji kryterialnej $\nabla\Psi_2(\mathbf{v})$ jest stały w pojedynczym obszarze korekcyjnym i można go łatwo policzyć jako ważoną sumę gradientów składowych funkcji kary, które wynoszą odpowiednio:

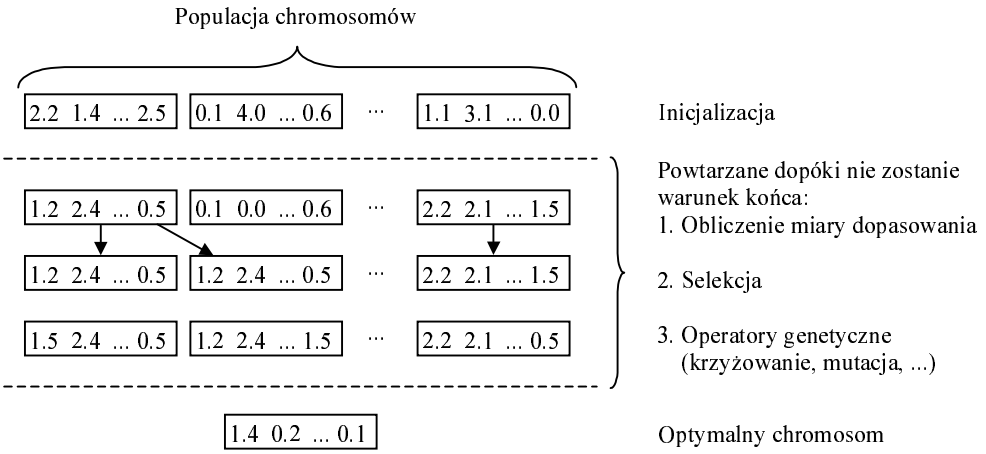
$$\nabla\varphi_j^+(\mathbf{v}) = \begin{cases} \mathbf{y}^j & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle < \delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \geq \delta^j \end{cases} \text{ oraz } \nabla\varphi_j^-(\mathbf{v}) = \begin{cases} -\mathbf{y}^j & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \geq -\delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle < -\delta^j \end{cases} \quad (13)$$

Po obliczeniu rzutów średniego wektora korekcji $\mathbf{k}(\mathbf{v})$ na hiperpłaszczyzny przecinające się w wierzchołku $\mathbf{v}(n)$ i związane z wektorami cech tworzącymi bazę wybierany jest kierunek, w którym ów rzut jest największy. Tym samym wyznaczamy wektor cech opuszczający bazę aktualną (*kryterium wyjścia*). Następnie poruszamy się w tym kierunku aż do osiągnięcia minimum w kolejnym wierzchołku $\mathbf{v}(n+1)$, co jest równoznaczne ze wskazaniem wektora wchodzącego do nowej bazy $\mathbf{B}(n+1)$ (*kryterium wejścia*). Kolejne kroki algorytmu są następnie powtarzane aż do osiągnięcia minimum globalnego (*kryterium stopu*).

4.2.2. Algorytm ewolucyjny

Algorytmy ewolucyjne (AE) są stochastycznymi technikami optymalizacji, które zostały zainspirowane przez proces ewolucji [17]. Ich największą zaletą w stosunku do metod gradientowych jest zdolność unikania optimów lokalnych a także możliwość wykorzystania specyfiki problemu podczas jego optymalizacji (np. poprzez zdefiniowanie specjalnych operatorów genetycznych czy odpowiednią reprezentację). Dzięki temu zostały one z powodzeniem zastosowane w wielu praktycznych zagadnieniach. W pewnym uproszczeniu zasada działania AE polega na utrzymywaniu populacji osobników (zwanych chromosomami), z których każdy koduje jedno rozwiązanie danego problemu. W kolejnych iteracjach algorytmu chromosomy ulegają zmianom wskutek działania operatorów genetycznych, przy czym osobniki lepiej przystosowane mają większą szansę na przeżycie i wydanie potomstwa. Jakość rozwiązań zakodowanych w poszczególnych chromosomach

jest oceniana na podstawie funkcji dopasowania (ang. *fitness function*). W sposób schematyczny AE przedstawiono na rysunku 6. W dalszej części paragrafu skoncentrowano się na specyficznych cechach algorytmu wykorzystanego do minimalizacji funkcji kryterialnej Ψ_1.



Rys. 6. Schemat działania algorytmu ewolucyjnego

Reprezentacja i inicjalizacja. Ponieważ pojedynczy chromosom ma kodować współczynniki hiperpłaszczyzny $H(\mathbf{v})$ najbardziej naturalna jest reprezentacja w postaci N+1 liczb rzeczywistych, która bezpośrednio odpowiada rozszerzonemu wektorowi wag $\mathbf{v}$. Do inicjalizacji populacji początkowej wykorzystano następujący prosty algorytm: dla każdego osobnika losujemy dwa wektory cech $\mathbf{x}^i$, $\mathbf{x}^j$ z różnych klas (dipol mieszany) i wybieramy wagi tak, aby $H(\mathbf{v})$ separowała te wektory:

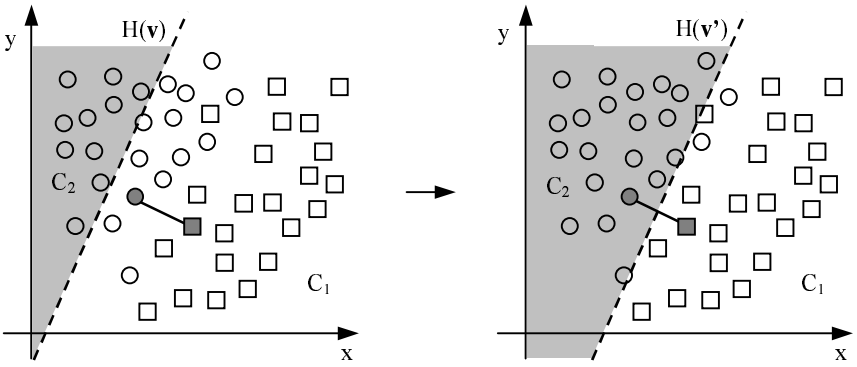
$$\mathbf{w} = \mathbf{x}^i - \mathbf{x}^j = \left[x_1^i - x_1^j, ..., x_N^i - x_N^j\right] \text{ oraz } \theta = \frac{1}{2}\left\langle \mathbf{w}, \mathbf{x}^i \right\rangle + \frac{1}{2}\left\langle \mathbf{w}, \mathbf{x}^j \right\rangle. \qquad (14)$$

Tak zdefiniowana hiperpłaszczyzna $H(\mathbf{v})$ leży w połowie odległości pomiędzy wektorami cech tworzącymi dipol $\left(\mathbf{x}^i, \mathbf{x}^j\right)$ i jest prostopadła do prostej przez nie przechodzącej.

Funkcja dopasowania. Algorytm ten służy do minimalizowania funkcji kryterialnej Ψ_1, w związku z czym jest ona wykorzystana do oceny przystosowania osobników.

Selekcja. Jako operator selekcji wykorzystana została proporcjonalna selekcja z liniowym skalowaniem, przy czym chromosom z najlepszą funkcją dopasowania jest zawsze kopiowany do kolejnej populacji (model elitarny).

Operatory genetyczne. Oprócz standardowego operatora krzyżowania arytmetycznego i mutacji wykorzystaliśmy dodatkowy specjalizowany operator genetyczny, który nazwany został operatorem *dipolowym*. Jego działanie jest następujące: losowany jest dipol i sprawdzane jest czy przecina go hiperpłaszczyzna $H(\mathbf{v})$. W zależności od typu dipola i od tego czy przemieszczenie jest w jego przypadku potrzebne dokonywana jest zmiana położenia $H(\mathbf{v})$ poprzez modyfikację pojedynczej wylosowanej cechy. Na rysunku 7 zilustrowano przykładowe działanie operatora w przypadku wylosowania dipola mieszanego, który nie jest w tej sytuacji przecinany. W prezentowanych eksperymentach prawdopodobieństwa wykorzystania operatorów wynosiły odpowiednio: krzyżowanie – 0,8, mutacja – 0,1, dipolowy – 0,5.



Rysunek 7. Przykład działania operatora dipolowego, który dla wylosowanego dipola mieszanego (który nie jest jeszcze przecięty) przemieszcza hiperpłaszczyznę tak, aby go przecięła

Warunek stopu. Algorytm zatrzymuje się, gdy przez pewną określoną liczbę iteracji funkcja dopasowania nie ulega poprawie bądź osiągnięta zostaje z góry założona maksymalna liczba iteracji (w prezentowanych eksperymentach wartości te wynosiły odpowiednio: 200 i 1000 iteracji).

5. Wielowymiarowe drzewa decyzyjne

Strategia generowania wielowymiarowego drzewa decyzyjnego wykorzystana w prezentowanym systemie nie odbiega od standardowo stosowanej rekurencyjnej metody „top-down" (opisana została w paragrafie 2), przy czym liniowe reguły decyzyjne w poszczególnych węzłach drzewa uzyskiwane są w wyniku zastosowania procedur opisanych w poprzednim paragrafie. Pozostałe kwestie związane z indukcją drzewa, takie jak selekcja cech oraz unikanie przetrenowania, zostaną przedstawione poniżej.

5.1. Selekcja cech

Użyteczność klasyfikatora zależy często nie tylko od jego skuteczności (mierzonej jakością klasyfikacji), ale również od możliwości zrozumienia jego działania i interpretacji sposobu proponowania decyzji. W takim przypadku dążenie do uproszczenia drzewa decyzyjnego jest jak najbardziej uzasadnione. Co więcej, w wielu sytuacjach, poprzez usunięcie z testów cech zaszumionych bądź nadmiarowych, uzyskujemy rozwiązania nie tylko prostsze, ale również bardziej poprawne. Dlatego też prawie wszystkie metody poszukiwania wielowymiarowych drzew decyzyjnych zostały wyposażone w mechanizm selekcji cech. Zwykle w każdym węźle dobór najbardziej użytecznych cech jest przeprowadzany niezależnie od pozostałych części drzewa[4]. W najbardziej optymistycznym przypadku może to prowadzić do testu wykorzystującego tylko pojedynczą cechę. Z drugiej strony musimy zdawać sobie sprawę, że problem wyboru optymalnego podzbioru atrybutów jest bardzo złożony (liczba możliwych podzbiorów rośnie wykładniczo wraz ze wzrostem liczby cech). W związku z tym stosowane w praktycznych zastosowaniach algorytmy selekcji cech są metodami heurystycznymi. W pracy [10] został przedstawiony przegląd metod selekcji cech w kontekście wielowymiarowych drzew decyzyjnych. Zdaniem autorów najlepsze wyniki można uzyskać stosując zaproponowaną przez nich metodę *Heuristic Sequential Search*. Jest to kombinacja dwu powszechnie znanych sekwencyjnych metod: wyboru (*Forward Selection*)
i eliminacji (*Backward Elimination*). Działa ona w ten sposób, że początkowo znajduje najlepsze testy składające się ze wszystkich cech oraz z pojedynczej cechy. Następnie porównuje je i, w zależności od wyniku, uruchamia standardową procedurę sekwencyjnego wyboru lub eliminacji.

[4] W niektórych bardziej specyficznych zastosowaniach (np. diagnostyce medycznej), gdzie poszczególne cechy mogą mieć przypisany (często bardzo różny) koszt wykorzystywane są podejścia oparte na powtórnym użyciu cech występujących w wyższych partiach drzewa.

Stosowanie w bezpośredni sposób powyższej metody okazało się nie być zbyt efektywne (w sensie złożoności czasowej), zwłaszcza w przypadku danych z dużą ilością cech [7]. Dlatego też opracowane zostały metody, w których selekcja cech została wbudowana w proces minimalizacji funkcji kryterialnej.

W przypadku algorytmu wymiany rozwiązań bazowych zaproponowane podejście polega na modyfikacji kryterium wyjścia (służącego do wyboru wektora cech, opuszczającego bazę) i krokowym uruchamianiu algorytmu. Należy przy tym wyjaśnić, że w momencie startu algorytmu wszystkie wagi są wyzerowane, w bazie znajdują się „sztuczne" wektory jednostkowe, a wprowadzenie wektora cech do bazy w wierszu *k*-tym równoznaczne jest z wykorzystaniem *k*-tej cechy. W pierwszym kroku wprowadzane są wektory cech tylko do dwu wierszy bazy (przy czym jeden odpowiada progowi $-\theta$) i jest to równoznaczne ze znalezieniem testu z pojedynczą cechą. Wybór wektorów cech opuszczających bazę odbywa się standardowo na podstawie wartości rzutu wektora korekcji. W kolejnych krokach dopuszczone jest wykorzystanie dodatkowo jednej cechy (jednego wiersza w bazie) i następuje sekwencja minimalizacji, w której wektory cech mogą opuszczać bazę tylko z tych miejsc. Każdorazowo po zakończeniu sekwencji minimalizacji otrzymujemy informację o wartości funkcji kryterialnej (bądź ilości przeciętych dipoli) odpowiadających rozpatrywanej liczbie cech. Dzięki takiemu mechanizmowi na podstawie pojedynczego przebiegu algorytmu wymiany rozwiązań bazowych możemy dokonać wyboru optymalnej liczby atrybutów tworzących hiperpłaszczyznę.

Jeśli chodzi o algorytm ewolucyjny, to minimalny zestaw zmian obejmuje przede wszystkim modyfikację funkcji dopasowania, która w przypadku wykorzystywania selekcji cech wygląda następująco:

$$Fitness(\mathbf{v}) = \Psi_1(\mathbf{v}) \cdot \left[(1-\alpha) + \alpha \frac{n}{N} \right] \tag{15}$$

gdzie:

n – liczba cech wykorzystana w teście,

α – parametr regulujący wpływ złożoności testu na jego ogólną ocenę ($\alpha \in 0..1$).

Tak zdefiniowana funkcja umożliwia premiowanie rozwiązań prostszych, a poprzez wartości α użytkownik ma możliwość wpływania na dążenie algorytmu do ograniczania liczby cech wykorzystanych w testach (we wszystkich eksperymentach $\alpha = 0{,}2$). Oprócz zmiany funkcji dopasowania niezbędne jest również ułatwienie usuwania cech z testu, zwłaszcza w przypadku zastosowanej reprezentacji rzeczywistej. Osiągnięte to zostało poprzez zmodyfikowanie mutacji, tak aby znacząco zwiększyć prawdopodobieństwo przypisania współczynnikom wartości 0, odpowiadającej wykluczeniu cechy z kombinacji liniowej.

Innym problem, który związany jest bezpośrednio z selekcją cech, jest zapewnienie odpowiedniej liczby wektorów cech w stosunku do liczby atrybutów [12]. Problem ten jest szczególnie istotny w dolnych partiach drzewa decyzyjnego („blisko liści"), gdzie wskutek wcześniejszych podziałów liczba dostępnych danych uczących w węźle ulega znacznej redukcji. Jeżeli liczba obiektów jest zbyt mała w stosunku do rozpatrywanej liczby cech, to nie możemy jednoznacznie wskazać optymalnej hiperpłaszczyzny, gdyż istnieje wiele równoważnych orientacji. Sytuacja ta może być określona mianem braku możliwości dopasowania do danych uczących (ang. *underfitting*). Aby jej uniknąć, należy ograniczyć rozpatrywaną liczbę cech w teście, tak aby liczba obiektów stanowiła wielokrotność liczby atrybutów (w prezentowanych eksperymentach przyjęto, że liczba wektorów cech musi być co najmniej 5-krotną wielokrotnością liczby cech).

5.2. Unikanie przetrenowania

W przypadku generowania klasyfikatorów na podstawie zbiorów danych, niezależnie od ich formy (dotyczy to w równym stopniu drzew decyzyjnych, zbiorów reguł decyzyjnych czy też sieci neuronowych), mamy często do czynienia z problemem unikania zbytniego dopasowania do danych (ang. *over-fitting*). Zwykle klasyfikator bardzo dobrze rozpoznaje przypadki wykorzystywane do uczenia, co niestety nie znaczy, że równie dobrze będzie sobie radził z danymi, których wcześniej nie poznał. Problem ten ma szczególne znaczenie podczas przetwarzania rzeczywistych, zwykle zaszumionych, zbiorów danych.

W odniesieniu do drzew decyzyjnych podstawowym parametrem kontrolującym ich zdolności generalizacyjne jest rozmiar (ilość węzłów i liści). Aby zapewnić odpowiedni rozmiar drzewa, stosowane są powszechnie dwie techniki [9]:

– przerwanie rekurencyjnego podziału obiektów w węzłach przed osiągnięciem perfekcyjnej klasyfikacji na zbiorze uczącym. Zdefiniowanie optymalnego kryterium stopu okazuje się jednak zagadnieniem bardzo trudnym.
– dodatkowe przycinanie (ang. *post-pruning*) drzewa uruchamiane po zakończeniu jego indukcji. Metoda ta jest oczywiście bardziej czasochłonna, tym niemniej eliminuje niepewność związaną ze zbyt wczesnym zatrzymaniem rekurencyjnego podziału i umożliwia dokładną analizę uzyskanego drzewa. Główna idea podcinania polega na zastępowaniu wybranego poddrzewa poprzez liść (lub ewentualnie jedną z gałęzi), o ile klasyfikuje on lepiej niż całe poddrzewo. Algorytmy przycinania rozpoczynają swoje działanie od najniższych partii drzewa, a następnie przesuwają w stronę korzenia, próbując zredukować coraz większe poddrzewa. Poszczególne algorytmy różnią się od siebie głównie sposobem wyznaczania jakości klasyfikacji rozpatrywanych części drzewa, przy czym warto zaznaczyć, że musi być ona określona inaczej niż poprzez reklasyfikację. W najprostszym przypadku wydziela się z danych uczą-

cych część wektorów cech, które nie są wykorzystywane podczas tworzenia drzewa, natomiast służą do oceny poddrzew w czasie przycinania. Metoda taka ma jedną podstawową wadę: ogranicza ilość danych służących do indukcji drzewa.

Przeprowadzono wiele badań eksperymentalnych, które jednoznacznie pokazują, że druga wymieniona powyżej technika daje lepsze rezultaty. Tymczasem najczęściej obie techniki są stosowane równocześnie. W prezentowanym systemie wykorzystana jest bardzo prosta reguła stopu – jeżeli liczba obiektów w węźle jest mniejsza niż M_{st}, to węzeł automatycznie staje się liściem bez względu na przynależność obiektów do klas (we wszystkich eksperymentach prezentowanych w pracy $M_{st} = 5$). W systemie zaimplementowano kilka spośród wielu algorytmów przycinania [13], tym niemniej we wszystkich eksperymentach wykorzystano zmodyfikowaną metodę „pesymistyczną", zaproponowaną przez Quinlana w systemie C4.5 [21]. Jest to metoda heurystyczna, która wyróżnia się tym, że nie wymaga podziału zbioru uczącego na części wykorzystywane do indukcji i walidacji.

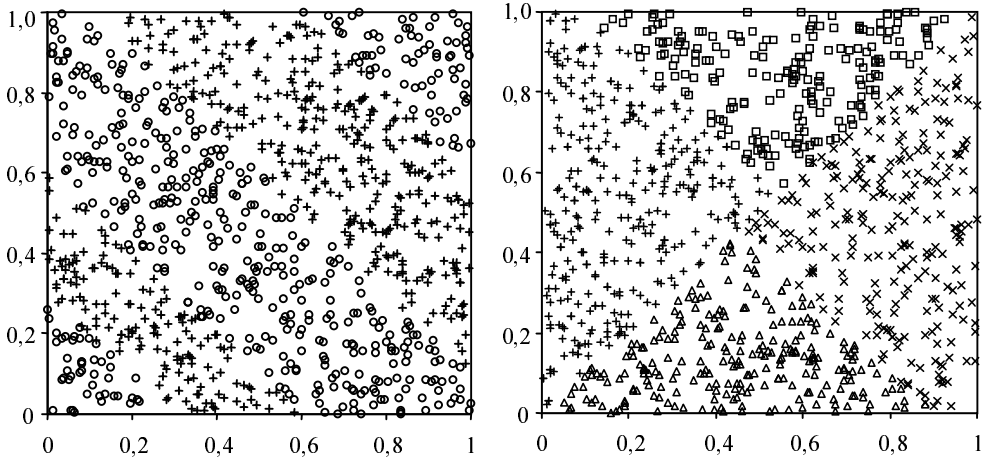
6. Wyniki eksperymentalne

Aby sprawdzić, na ile zaproponowane algorytmy spełniają swoje zadanie, przeprowadzono dwie serie eksperymentów. W pierwszej operowano na sztucznie wygenerowanych zbiorach danych, dla których jednoznacznie zdefiniowano podział na klasy i dzięki temu znane jest rozwiązanie optymalne. Druga seria eksperymentów została przeprowadzona na rzeczywistych zbiorach danych, obejmujących szeroki zakres problemów praktycznych pochodzących z repozytorium Uniwersytetu Kalifornijskiego w Irwine [1]. Zbiory tam zgromadzone są często wykorzystywane przy porównywaniu algorytmów tworzących wszelkiego typu klasyfikatory. Z jednego z takich porównań [16] zaczerpnięte zostały wyniki uzyskane przez inne systemy (C4.5 i OC1).

Wszystkie prezentowane w tej pracy rezultaty uzyskano przy wykorzystaniu bądź powtórzonej 5-krotnie kroswalidacji 10-przedziałowej, bądź zbioru testowego (o ile był dostępny). Złożoność drzew mierzona jest ilością liści, natomiast podawane czasy dotyczą pojedynczego uruchomienia algorytmów indukcji drzew przy wykorzystaniu całego zbioru uczącego. Obliczenia wykonano na komputerze klasy PC (PII 350MHz). W prezentowanych poniżej tabelach skrót MDT-EA odnosi się do prostszej metody wykorzystującej algorytm ewolucyjny do minimalizacji funkcji kryterialnej, natomiast MDT-BEA odpowiada metodzie opartej na algorytmie wymiany rozwiązań bazowych.

6.1. Zbiory syntetyczne

Wszystkie zbiory analizowane w tym podpunkcie zbudowane są z 2000 wektorów cech należących do 2 klas. Cechy przyjmują wartości rzeczywiste z przedziału [0,1]. Dwa pierwsze zbiory prezentują analogiczny liniowo separowalny przypadek (stąd ich nazwa – LS), a różnią się liczbą wykorzystanych zmiennych (odpowiednio 2 i 10 atrybutów). Hiperpłaszczyzny decyzyjne rozdzielające klasy są zdefiniowane następująco: $x_1 < x_2$ oraz $x_1 + x_2 + x_3 + x_4 + x_5 < x_6 + x_7 + x_8 + x_9 + x_{10}$. Dwa kolejne zbiory są 2-wymiarowe i zostały przedstawione na rysunku 7. Analogiczne testy na syntetycznych zbiorach danych przeprowadzone zostały w pracy [18]. Oryginalne zbiory nie są, niestety publicznie dostępne, stąd bezpośrednie porównanie uzyskanych wyników nie jest możliwe.



Rys. 7. Wykresy zbiorów syntetycznych: „Zebra" i „Szachownica"

Tabela 2

Porównanie na syntetycznych zbiorach danych

Zbiór danych	MDT-EA			MDT-BEA		
	Jakość kl.[%]	Złożoność	Czas [s]	Jakość kl. [%]	Złożoność	Czas [s]
LS2	99,85	2	10,5	99,85	2	8,5
LS10	98,30	8	19,8	99,55	2	180
„Zebra"	99,35	6	23,2	99,35	9	25,2
„Szachownica"	99,55	6	16,9	99,25	13	36,0

W przypadku pierwszego zbioru (LS2) oba algorytmy poradziły sobie bez trudu, znajdując optymalne rozwiązanie. Przeniesienie problemu na większą liczbę wymiarów pokazało, że zagadnienie to przestało być trywialne. Prostsza metoda oparta na algorytmie ewolucyjnym, odnalazła podstawowy podział, tym niemniej pojawiły się problemy z dokładnym dopasowaniem i stąd spory wzrost złożoności. Podobnie zachowuje się OC1 (na podstawie [18]), który uzyskał 97,2% przy 14 liściach. Jeśli chodzi o dwa kolejne zbiory, to pierwszy algorytm odnalazł rozwiązania prawie optymalne, natomiast drugi, zgodnie z oczekiwaniami, miał nieco większe problemy. Wpływ dipoli mieszanych utworzonych z dalekich dipoli spowodował przesunięcia hiperpłaszczyzn i w związku z tym nie udało się osiągnąć optymalnej złożoności (odpowiednio 5 i 4 liście), tym niemniej jakość klasyfikacji jest bardzo wysoka.

6.2. Zbiory rzeczywiste

Tabela 4 zawiera charakterystykę rzeczywistych zbiorów danych wykorzystanych w drugiej serii eksperymentów, natomiast tabele 5 i 6 przedstawiają wyniki uzyskane przez poszczególne algorytmy (odpowiednio jakość klasyfikacji i złożoność drzew w 5 i czas obliczeń w 6). Czasy obliczeń podane są wyłącznie dla algorytmów „dipolowych", gdyż nie dysponujemy analogicznymi danymi dla pozostałych algorytmów.

Tabela 4

Charakterystyka rzeczywistych zbiorów danych

Zbiór danych	Liczba obiektów (część testowa)	Liczba cech	Liczba klas
Breast (Wisconsin)	683	9	2
CMC	1473	9	3
Heart (Cleveland)	270	13	2
Pima	532	7	2
Segmentation	2310	19	7
Smoking	1855(1000)	8	3
Vehicle	846	18	4
Waveform	600(3000)	21	3

Tabela 5

Jakość klasyfikacji i złożoności drzew uzyskane przez poszczególne algorytmy

Zbiór danych	Jakość klasyfikacji [%]				Złożoność drzew [liczba liści]			
	MDT-EA	MDT-BEA	C4.5	OC1	MDT-EA	MDT-BEA	C4.5	OC1
Breast	95,2	96,6	95,8	95,9	11	4	11	5
CMC	49,0	53,0	51,7	52,2	243	129	143	8
Heart	78,2	80,6	80,4	77,8	20	10	23	3
Pima	72,4	74,7	75,8	75,3	49	14	18	5
Segmentation	95,1	96,4	96,8	94,3	51	28	42	26

Smoking	60,5	71,1	69,5	69,5	122	23	1	2
Vehicle	66,9	76,2	72,3	68,4	88	36	65	41
Waveform	77,5	80,0	73,9	78,2	39	25	54	15

Uzyskane wyniki pokazują, że metoda generowania drzewa oparta na algorytmie wymiany rozwiązań bazowych (MDT-BEA) poradziła sobie ze zbiorami rzeczywistymi najlepiej spośród analizowanych systemów, uzyskując najwyższą jakość klasyfikacji w 6 na 8 zbiorów. Wiązało się to jednak ze wzrostem złożoności tworzonych klasyfikatorów. Interesujące jest również, że w porównaniu do poprzednio stosowanej selekcji cech [7] udało się ograniczyć czas obliczeń od 3 do 5 razy w zależności od zbioru danych, przy zachowaniu bardzo zbliżonych wyników w sensie jakości klasyfikacji. Zgodnie z oczekiwaniami, prostsza z metod dipolowych, oparta na algorytmie ewolucyjnym (MDT-EA) poradziła sobie tym razem gorzej. Zarówno jakość klasyfikacji jak i złożoność uzyskiwanych w ten sposób drzew decyzyjnych ustępują pozostałym systemom.

Tabela 6

Czas potrzebny do wygenerowania wielowymiarowego drzewa decyzyjnego na podstawie całego zbioru uczącego (w sekundach)

Zbiór danych	MDT-EA	MDT-BEA
Breast	28	10
CMC	192	82
Heart	29	8
Pima	38	16
Segmentation	453	697
Smoking	175	70
Vehicle	197	115
Waveform	98	66

7. Podsumowanie i planowane badania

W pracy przedstawiono algorytmy generowania wielowymiarowych drzew decyzyjnych przy wykorzystaniu dipolowych funkcji kryterialnych. Opierając się na dotychczasowych wynikach badań można stwierdzić, że zaproponowana metoda (wykorzystująca algorytm wymiany rozwiazań bazowych) charakteryzuje się jakością działania co najmniej porównywalną z najbardziej znanymi systemami tworzącymi klasyfikatory tego typu. Należy jednak zauważyć, że istnieje jeszcze cały szereg możliwych usprawnień, które zdaniem autorów powinny wpłynąć zarówno na poprawę jakości klasyfikacji jak i na zmniejszenie złożoności klasyfikatora. Przykładowo możliwe jest zaproponowanie nowych strategii wyceny dipoli a także, co się z tym bezpośrednio wiąże, wykorzystanie kryterium rangowego

alternatywnie, obok kryterium dipolowego, w tym samym węźle. Rozważane jest wprowadzenie dodatkowego kroku polegającego na lokalnym dopasowaniu uzyskanej hiperpłaszczyzny, mające na celu wyeliminowanie możliwego negatywnego wpływu dalekich wektorów cech. Planowane jest również opracowanie wersji rozproszonej algorytmu, która powinna umożliwić przetwarzanie znacznie większych zbiorów danych w sensownym czasie.

Przedstawione w pracy metody mogą być stosowane w różnorodnych zagadnieniach praktycznych. Obecnie trwają prace nad ich zastosowaniem w rozpoznawaniu zmian patologicznych wątroby na obrazach tomografii komputerowej, gdzie jako cechy wykorzystywane są charakterystyki teksturalne, obliczone dla analizowanych obszarów.

Referencje:

[1] Blake, C., Merz, C., (1998) *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html] Irvine, CA: University of California, Department of Information and Computer Science.

[2] Bobrowski, L., (1991) *Design of piecewise linear classifiers from formal neurons by some basis exchange technique*, Pattern Recognition, 24(9): 862-870.

[3] Bobrowski, L., (1996) *Piecewise-linear classifiers, formal neurons and separability of the learning sets*, In Proc. of Int. Conf. on Pattern Recognition ICPR'96 (Vienna), 224-228.

[4] Bobrowski, L., (1999) *Data mining procedures related to the dipolar criterion function*, Applied Stochastic Models and Data Analysis – Quantitative Methods in Business and Industry Society, INE (Lisboa), 43-50.

[5] Bobrowski, L., (2000) *Strategie projektowania sieci neuropodobnych,* Biocybernetyka i Inżynieria Biomedyczna 2000, Tom 6, Akademicka Oficyna Wydawnicza EXIT, 295-321.

[6] Bobrowski, L., Krętowska, M., Krętowski, M., (1998) *Generowanie sieci neuropodobnych oraz drzew decyzyjnych w oparciu o kryteria dipolowe*, Symulacja w badaniach i rozwoju, 17-28.

[7] Bobrowski, L., Krętowski M., (2000) *Induction of Multivariate Decision Trees by using Dipolar Criteria*, Principles of Data Mining and Knowledge Discovery. 4th European Conference, PKDD'2000. Springer LNCS 1910, 331-336.

[8] Bobrowski, L., Niemiro, W., (1984) *A method of synthesis of linear discriminant function in the case of nonseparability*, Pattern Recognition, 17: 205-210.

[9] Breiman, L., Friedman, J., Olshen, R., Stone C., (1984) *Classification and Regression Trees*, Wadsworth International Group.

[10] Brodley, C., Utgoff, P., (1995) *Multivariate decision trees*, Machine Learning, 19: 45-77.
[11] Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J., (1996) *Piecewise-linear classifiers using binary tree structure and genetic algorithm*, Pattern Recognition, 29(11): 1905-1917.
[12] Duda, O.R., Heart, P.E., Stork D.G., (2001) *Pattern Classification*, Wydanie drugie, zmienione, John Wiley & Sons.
[13] Esposito, F., Malerba, D., Semeraro, G., (1997) *A comparative analysis of methods for pruning decision trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5): 476-491.
[14] Gama, J., Brazdil, P., (1999) *Linear tree*, Inteligent Data Analysis, 3(1): 1-22.
[15] Kass, G. V., (1980) *An exploratory technique for investigating large quantities of categorical data*, Applied Statistics, 29(2): 119-127.
[16] Lim, T., Loh, W., Shih, Y., (2000) *A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms*, Machine Learning 40(3): 203-228.
[17] Michalewicz, Z., (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer.
[18] Murthy, S., Kasif, S., Salzberg, S., (1994) *A system for induction of oblique decision trees*, Journal of Artificial Intelligence Research, 2: 1-33.
[19] Murthy, S., (1998) *Automatic construction of decision trees from data: A multi-disciplinary survey*, Data Mining and Knowledge Discovery, 2:345-389.
[20] Shah, S., Sastry, P.S., (1999) *New algorithms for learning and pruning oblique decision trees*, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 29(4): 494-505.
[21] Quinlan, J., (1986) *Induction of decision trees.* Machine Learning 1(1): 81-106.
[22] Quinlan, J., (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers.
[23] Utgoff, P., Brodley, C., (1991) *Linear machine decision trees*, (COINS Technical report 91-10), Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

INDUCTION OF MULTIVARIATE DECISION TREES FROM DATASETS

Abstract: In the paper, new methods for induction of classifiers in the form of multivariate (oblique) decision trees are presented. A linear decision function is used at each non-terminal node of the binary tree for splitting the data. The strategies for finding the decision hyper-plane are based on the concept of dipoles and as optimization procedures they

use evolutionary algorithms or basis exchange algorithms. To eliminate redundant and noisy features we embedded into the process of searching for the optimal linear combination in each node the feature selection mechanism. To avoid over-fitting and to increase generalization the tree is pruned back after the growing phase. The presented methods were experimentally verified on publicly available datasets and compared with other decision tree systems.

Keywords: multivariate (oblique) decision trees, linear decision function, dipolar criteria

Artykuł zrealizowano w ramach pracy badawczej W/II/1/00.